

# **Отчёт по лабораторной работе №8**

**Шифр гаммирования**

**Кузьмин Артем Дмитриевич**

# Содержание

<b>1 Цель работы.....</b>	<b>3</b>
<b>2 Выполнение работы.....</b>	<b>6</b>
<b>3 Выводы.....</b>	<b>9</b>

## **1 Цель работы**

Освоить на практике применение режима однократного гаммирования на примере кодирования различных исходных текстов одним ключом.

## 2 Теоретические сведения

### 2.1 Шифр гаммирования

Гаммирование – это наложение (снятие) на открытые (зашифрованные) данные криптографической гаммы, т.е. последовательности элементов данных, вырабатываемых с помощью некоторого криптографического алгоритма, для получения зашифрованных (открытых) данных.

Принцип шифрования гаммированием заключается в генерации гаммы шифра с помощью датчика псевдослучайных чисел и наложении полученной гаммы шифра на открытые данные обратимым образом (например, используя операцию сложения по модулю 2). Процесс дешифрования сводится к повторной генерации гаммы шифра при известном ключе и наложении такой же гаммы на зашифрованные данные. Полученный зашифрованный текст является достаточно трудным для раскрытия в том случае, если гамма шифра не содержит повторяющихся битовых последовательностей и изменяется случайным образом для каждого шифруемого слова. Если период гаммы превышает длину всего зашифрованного текста и неизвестна никакая часть исходного текста, то шифр можно раскрыть только прямым перебором (подбором ключа). В этом случае криптостойкость определяется размером ключа.

Метод гаммирования становится бессильным, если известен фрагмент исходного текста и соответствующая ему шифрограмма. В этом случае простым вычитанием по модулю 2 получается отрезок псевдослучайной последовательности и по нему восстанавливается вся эта последовательность.

Метод гаммирования с обратной связью заключается в том, что для получения сегмента гаммы используется контрольная сумма определенного участка шифруемых данных. Например, если рассматривать гамму шифра как объединение непересекающихся множеств  $H(j)$ , то процесс шифрования можно представить следующими шагами:

1. Генерация сегмента гаммы  $H(1)$  и наложение его на соответствующий участок шифруемых данных.
2. Подсчет контрольной суммы участка, соответствующего сегменту гаммы  $H(1)$ .
3. Генерация с учетом контрольной суммы уже зашифрованного участка данных следующего сегмента гамм  $H(2)$ .
4. Подсчет контрольной суммы участка данных, соответствующего сегменту данных  $H(2)$  и т.д.

### 2.2 Идея взлома

Шифротексты обеих телеграмм можно получить по формулам режима однократного гаммирования:

$$C1 = P1 \oplus K$$

$$C2 = P2 \oplus K$$

Открытый текст можно найти, зная шифротекст двух телеграмм, зашифрованных одним ключом. Для это оба равенства складываются по модулю 2. Тогда с учётом свойства операции XOR получаем:

$$C1 \oplus C2 = P1 \oplus K \oplus P2 \oplus K = P1 \oplus P2$$

Предположим, что одна из телеграмм является шаблоном — т.е. имеет текст фиксированный формат, в который вписываются значения полей. Допустим, что злоумышленнику этот формат известен. Тогда он получает достаточно много пар  $C1 \oplus C2$  (известен вид обеих шифровок). Тогда зная  $P1$  имеем:

$$C1 \oplus C2 \oplus P1 = P1 \oplus P2 \oplus P1 = P2$$

Таким образом, злоумышленник получает возможность определить те символы сообщения  $P2$ , которые находятся на позициях известного шаблона сообщения  $P1$ . В соответствии с логикой сообщения  $P2$ , злоумышленник имеет реальный шанс узнать ещё некоторое количество символов сообщения  $P2$ . Затем вновь используется равенство с подстановкой вместо  $P1$  полученных на предыдущем шаге новых символов сообщения  $P2$ . И так далее. Действуя подобным образом, злоумышленник даже если не прочитает оба сообщения, то значительно уменьшит пространство их поиска.

## 3 Выполнение работы

### 3.1 Реализация взломщика, шифратора и дешифратора на Python

```
a = ord("a")
litters = [chr(i) for i in range(a, a + 32)]
a = ord("0")
for i in range(a, a+10):
    litters.append(chr(i))

a = ord("A")
for i in range(1040, 1072):
    litters.append(chr(i))
P1 = "КодофаяФраза1"
P2 = "Безопасность2"
def vzlom(P1, P2):
    code = []
    for i in range(len(P1)):
        code.append(litters[(litters.index(P1[i]) + litters.index(P2[i])) % len(litters)])
    print(code)
    pr = "".join(code)
    print(pr)
def shifr(P1, gamma):
    dicts = {"a": 1, "б": 2, "в": 3, "г": 4, "д": 5, "е": 6, "ё": 7, "ж": 8, "з": "м": 14, "н": 15, "о": 16, "п": 17,
    "р": 18, "с": 19, "т": 20, "у": "ш": 26, "щ": 27, "ь": 28, "ы": 29, "ъ": 30, "э": 31, "ю": 32, "я": "д": 37,
    "Е": 38, "Ё": 39, "Ж": 40, "З": 41, "И": 42, "Й": 43, "К": 44, "Л": 49, "Р": 50, "С": 51, "Т": 52, "У": 53,
    "Ф": 54, "Х": 55, "Ц": 5, "Ы": 61, "Ь": 62, "Э": 63, "Ю": 64, "Я": 65, "1": 66, "2": 67, "3": 6, "8": 73, "9": 74,
    "0": 75 }
    dicts2 = {v: k for k, v in dicts.items()}
    text = P1
    digits_text = []
    digits_gamma = []
    for i in text:
        digits_text.append(dicts[i])
        print("Числа текста ", digits_text)
```

```

    for i in gamma:
        digits_gamma.append(dict2[i])
    print("Числа гаммы ", digits_gamma)
    digits_result = []
    ch = 0
    for i in text:
        try:
            a = dict2[i] + digits_gamma[ch]
        except:
            ch = 0
            a = dict2[i] + digits_gamma[ch]
            if a > 75:
                a = a % 75
            print(a)
            ch += 1
            digits_result.append(a)
    print("Числа шифротекста ", digits_result)

text_cr = ""
for i in digits_result:
    text_cr += dict2[i]
print("Шифротекст ", text_cr)
digits = []
for i in text_cr:
    digits.append(dict2[i])
ch = 0
digits1 = []
for i in digits:
    try:
        a = i - digits_gamma[ch]
    except:
        ch = 0
        a = i - digits_gamma[ch]
    if a < 1:
        a = 75 + a
    digits1.append(a)

```

```

ch += 1
text_decr = ""
for i in digits1:
    text_decr += dicts2[i]
print("Расшифрованный текст: ", text_decr)

```

## 3.2 Контрольный пример

```

13
14 def vzlom(P1, P2):
15     code = []
16     for i in range(len(P1)):
17         code.append(liters[(liters.index(P1[i]) + liters.index(P2[i])) % len(liters)])
18     print(code)
19     pr = "".join(code)
20     print(pr)

```

In [11]: 1 len(P1)

Out[11]: 13

In [12]: 1 len(P2)

Out[12]: 13

In [13]: 1 vzlom(P1, P2)

['х', 'у', 'л', 'ь', 'з', 'а', 'ж', 'б', 'ю', 'с', 'щ', 'ь', 'щ']  
хульЗаЖбюсцьЩ

Рис. 1: Работа алгоритма взлома ключа

```

In [24]: 1 P1 = "КодофаяФраза1"
          2 gamma = "хульЗаЖбюсцьЩ"

```

In [25]: 1 shifr(P1, gamma)

Числа текста [44, 16, 5, 16, 22, 1, 32, 54, 18, 1, 9, 1, 66]  
Числа гаммы [23, 21, 13, 30, 68, 1, 40, 2, 32, 19, 27, 30, 59]  
15  
50  
Числа шифротекста [67, 37, 18, 46, 15, 2, 72, 56, 50, 20, 36, 31, 50]  
Шифротекст 2ДрМн67ЦРтГэР  
Расшифрованный текст: КодофаяФраза1

Figure 2: Работа алгоритма шифрования и дешифровки



## **4 Вывод**

В ходе выполнения лабораторной работы было разработано приложение, позволяющее шифровать тексты в режиме однократного гаммирования